Zwinność i dyscyplina w podnoszeniu efektywności zespołów projektowych

T. GÓRSKI
e-mail: gorski@wat.edu.pl

Instytut Systemów Informatycznych
Wydział Cybernetyki WAT
ul. S. Kaliskiego 2, 00-908 Warszawa

W artykule przedstawiono założenia metodyki projektowania systemów informatycznych łączącej cechy Rational Unified Process oraz metodyk zwinnych, takich jak SCRUM i OpenUP. Scharakteryzowane zostały metodyki Rational Unified Process, SCRUM i OpenUP z uwypukleniem ich cech wspólnych. Artykuł zawiera także analizę wyników zastosowania proponowanej metodyki w projekcie informatycznym w kontekście podnoszenia efektywności zespołu projektowego pracującego zgodnie z tą metodyką.

Słowa kluczowe: efektywność procesu projektowania, architektura systemu informatycznego

1. Podejścia projektowe

Zwinne podejście do projektowania systemów informatycznych określa, że istotne elementy w projektowaniu tego typu systemów to [9]:

- procesy i narzędzia
- szczegółowa dokumentacja
- negocjacje kontraktowe
- przestrzeganie planu.

Natomiast, zgodnie z założeniami podejścia zwinnego, jeszcze bardziej istotnymi elementami są:

- osoby i interakcje
- funkcjonujące oprogramowanie
- współpraca z klientem
- dostosowywanie się do zmian.

Widać z tego, że podejście zwinne przenosi punkt skupienia z procesu z przypisanymi rolami, sztywnego trzymania się szczegółowego planu i nadmiernego dokumentowania decyzji projektowych na jak najszybsze uzyskiwanie funkcjonującego oprogramowania spełniającego zmieniające się potrzeby klienta.

W ocenie autora nie oznacza to jednak odrzucenia procesu projektowania i określonych ról, gdyż role są zdefiniowane w metodykach zwinnych, lecz koncentrację na uzyskaniu efektu końcowego, jakim jest funkcjonujące zgodnie z potrzebami klienta oprogramowanie.

Praktyka projektowa pokazuje, że klienci podpisując kontrakt, mają jasno zdefiniowane cele tworzenia systemu. Oprogramowanie to ma określony zakres funkcjonalności, którą należy zbudować w określonym czasie i budżecie. Dwa ostatnie elementy w trakcie trwania projektu nie ulegają wydłużeniu ani zwiększeniu.

Klient podpisując kontrakt, oczekuje określonych korzyści biznesowych, jakie przyniesie mu wdrożenie oprogramowania, które zamawia. Musi ono spełniać określone wymagania funkcjonalne oraz, co bardzo istotne, pozafunkcjonalne. W szczególności wymagania związane z wydajnością i bezpieczeństwem.

Zdaniem autora, najlepszy efekt uzyskuje się, łącząc najlepsze cechy metodyk tradycyjnych, takich jak IBM Rational Unified Process oraz metodyk zwinnych, takich jak OpenUP czy SCRUM. W ten sposób można uzyskać proces projektowania systemu informatycznego, który zapewni odpowiedni poziom dyscypliny w prowadzeniu projektu z określonymi terminami budżetem oraz adaptowalność zespołu projektowego do zmieniających się w trakcie projektu wymagań. Przy bliższym spojrzeniu na te metodyki okazuje się, że nie są one od siebie tak odległe jak mogłoby się wydawać.

2. Rational Unified Process

Rational Unified Process (RUP) jest iteracyjną metodą wytwarzania oprogramowania. Metodyka RUP ma opinię ciężkiej metodyki. Argumentuje się to olbrzymim nakładem prac na wytwarzanie dokumentacji projektu. Według niektórych [11] prace nad dokumentacją zajmują do 50% wszystkich prac projektowych. Tego typu wyliczenia wskazują na stosowanie metodyki RUP w pełnej postaci, co jest pomysłem, delikatnie mówiąc, nierozsądnym.

U podstaw RUP leży sześć kluczowych reguł projektowania systemów informatycznych ukierunkowanego na zastosowanie biznesowe:

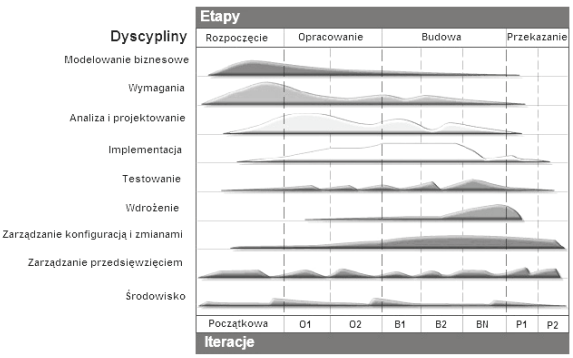
- dostosowanie procesu
- bilansowanie przeciwstawnych priorytetów udziałowców
- współpraca między zespołami
- demonstrowanie wartości w sposób iteracyjny
- dostosowanie poziomu abstrakcji
- ciągłe skupienie na jakości.

Za regułami tymi kryją się podstawowe dla inżynierii oprogramowania praktyki:

- wytwarzaj iteracyjnie
- zarządzaj wymaganiami
- używaj architektury komponentowej
- modeluj wizualnie (UML2.0)
- stale weryfikuj jakość
- zarządzaj zmianami.

Na podkreślenie zasługują praktyki związane z zarządzaniem wymaganiami, modelowaniem wizualnym z UML oraz używaniem architektury komponentowej.

Struktura procesu RUP jest dwuwymiarowa, co przedstawia rysunek 1. Wymiar poziomy pokazuje ułożenie projektu w czasie z podziałem na etapy i iteracje. Wymiar pionowy ukazuje dyscypliny procesu RUP. W ramach RUP występuje dziewięć dyscyplin − począwszy od modelowania biznesowego, a skończywszy na utrzymaniu środowiska produkcyjnego. W każdej z dyscyplin określone są czynności i role odpowiedzialne za wykonanie tych czynności, produkty potrzebne do realizacji czynności, a także produkty będące wynikiem realizacji czynności [6].



Rys. 1. Struktura RUP

Z punktu widzenia planowania projektu istotne są dwa plany: *plan przedsięwzięcia* i *plan iteracji*, za które odpowiada kierownik projektu. *Plan przedsięwzięcia* zawiera takie informacje, jak terminy głównych kamieni milowych faz, profil zespołu przedsięwzięcia, terminy pomniejszonych kamieni milowych. *Plan iteracji* jest planem szczegółowym dotyczącym aktualnej iteracji. *Plan iteracji* zawiera terminy iteracji, jej zakres oraz kryteria odbioru. Podczas każdej iteracji realizowane są czynności związane z dyscyplinami potrzebnymi do jej realizacji. Udział czynności z poszczególnych dyscyplin zmienia się wraz z postępem procesu.

Czas trwania iteracji nie jest ściśle określony. RUP rekomenduje, że iteracja powinna trwać od dwóch do sześciu tygodni [8]. Planowanie iteracji zgodnie z metodyką RUP powinno składać się z czterech kroków:

- określenie zakresu iteracji
- określenie kryteriów oceny iteracji
- określenie czynności wykonanych podczas iteracji
- przypisanie odpowiedzialności do poszczególnych czynności.

Cele iteracji zależą od etapu, w którym rozpatrywana iteracja jest przeprowadzana. Pierwszym etapem w metodyce RUP jest etap rozpoczęcia. Głównymi celami etapu rozpoczęcia są określenie zakresu sytemu, przedstawienie architektury kandydującej systemu, określenie listy ryzyk.

W etapie opracowania na definiowanie głównymi celami są zweryfikowanie architektury systemu oraz rozwiązanie głównych ryzyk. Szczegółowemu opisowi i implementacji w etapie opracowania powinny podlegać te przypadki użycia, które są krytyczne z punktu widzenia architektury oraz te, które realizują funkcjonalność najistotniejszą dla udziałowców.

Celem etapu budowy jest wytworzenie stabilnej wersji systemu z pełną wymaganą funkcjonalnością. Etap budowy jest najbardziej pracochłonnym z etapów i zajmuje przeciętnie 50% czasu trwania całego projektu.

Celem etapu przekazania jest dostarczenie końcowej wersji produktu użytkownikowi.

Metodyka Rational Unified Process uważana jest za metodykę ciężką ze względu na tworzoną dużą liczbę dokumentów. Dotyczy to jednak pełnej metodyki RUP, która nie powinna być w takiej postaci stosowana w projektach. Należy pamiętać o pierwszej zasadzie RUP, tj. o dostosowaniu procesu i doborze poziomu dokumentowania procesu do wielkości i złożoności projektu.

3. SCRUM

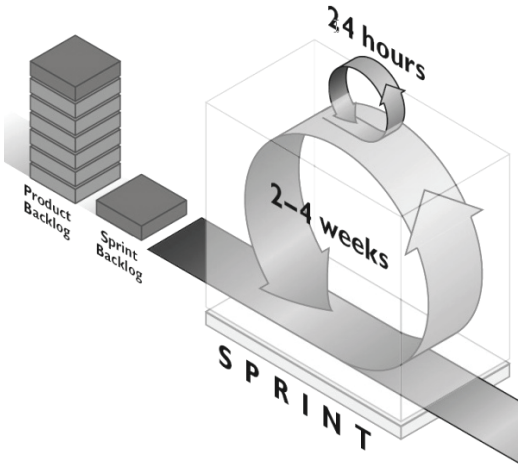
Scrum to, zgodnie z Agile Manifesto, zwinna metodyka prowadzenia projektów. Najczęściej jest ona wykorzystywana w projektach informatycznych. SCRUM używany jest w projektach charakteryzujących się wysokim poziomem zmienności wymagań lub w przypadku przedsięwzięć o wysokim stopniu innowacyjności. Metodyka skupia się na:

- dostarczaniu kolejnych, coraz bardziej dopracowanych wyników projektu
- włączaniu się przyszłych użytkowników w proces wytwórczy
- samoorganizacji zespołu projektowego.

Zazwyczaj zespół SCRUM składa się z 5-9 osób. Dobrze, gdy ma on charakter interdyscyplinarny i składa się z osób reprezentujących różne umiejętności. Osoby uczestniczące w zespole nie mogą uczestniczyć w innych zespołach. Główne role w projekcie to: mistrz młyna (ang. Scrum Master), właściciel produktu (ang. Product Owner) i członkowie zespołu (ang. The Team).

Zespół projektowy pracuje w określonym przedziale czasowym zwanym przebiegiem (ang. *sprint*). Efektem przebiegu za każdym razem powinno być dostarczenie użytkownikom kolejnego działającego produktu. Zasadą jest to, że prace wykonywane w ramach przebiegu muszą skutkować dostarczeniem nowej funkcjonalności. Przebieg może trwać od 2 do 6 tygodni. Zaleca się stosowanie przebiegów o stałych długościach.

W pierwszym etapie tworzona jest lista wymagań użytkownika. Właściciel projektu jest też zobowiązany do przedstawienia priorytetów wymagań oraz głównego celu przebiegu. Po tym formułowany jest rejestr wymagań (ang. Product Backlog). Następnie wybierane są zadania o najwyższym priorytecie, a jednocześnie przyczyniające się do realizacji celu projektu. Szacuje się czas realizacji każdego zadania. Lista zadań wraz z oszacowaną czasochłonnością nosi nazwę rejestru zadań przebiegu (ang. Sprint Backlog). W trakcie realizacji przebiegu Właściciel Produktu nie może ingerować w prace zespołu. Nie powinno się także zmieniać zakresu przebiegu. Przebieg prac zgodnie ze SCRUM został przedstawiony na rysunku 2.



Rys. 2. Przebieg prac zgodnie ze SCRUM [13]

Zespół z założenia jest samoorganizujący się i nie ma odgórnego przypisywania zadań do poszczególnych członków zespołu. Samodzielnie dokonują oni wyboru realizowanych zadań, według wspólnych ustaleń i umiejętności.

Naczelną zasadą metodyki jest przeprowadzanie codziennych (około piętnastominutowych) spotkań (ang. scrum meeting), na których omawiane są zadania zrealizowane poprzedniego dnia i problemy występujące przy ich realizacji oraz zadania do wykonania w dniu spotkania.

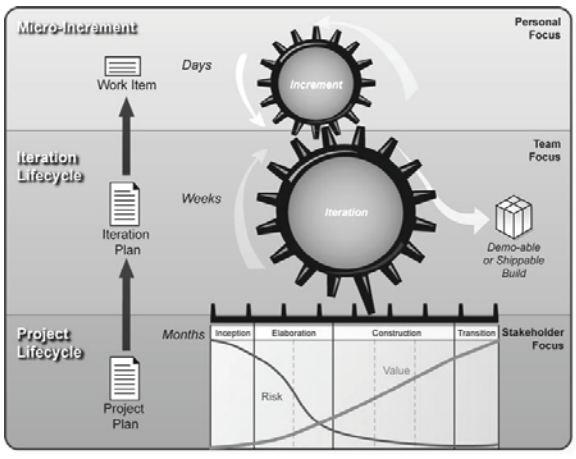
Przebieg kończy się przeglądem przebiegu (ang. *sprint review*), na którym prezentowany jest wynik pracy zespołu przez prezentowanie produktu wykonanego podczas przebiegu. Powinni w nim uczestniczyć wszyscy zainteresowani projektem. Po omówieniu produktu ustalany jest termin spotkania planistycznego do następnego przebiegu.

4. OpenUP

Metodyka Open Unified Process, w skrócie OpenUP, jest częścią projektu Eclipse Process Framework (EPF). OpenUP dostarcza zbiór dobrych praktyk iteracyjnego wytwarzania oprogramowania. OpenUP proponuje podzielić cały proces wytwarzania oprogramowania na trzy następujące obszary:

- cykl życia projektu (ang. Project Lifecycle)
- cykl życia iteracji (ang. Iteration Lifecycle)
- mikroprzyrosty (ang. Micro-Increments).

Podział ten oraz jego najważniejsze aspekty przedstawia rysunek 3.



Rys. 3. Obszary OpenUP [2]

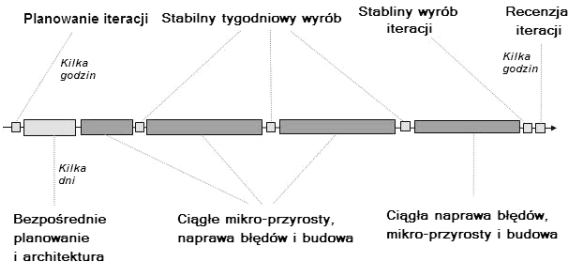
Każdy z obszarów posiada artefakt, który opisuje jego zakres oraz prace do wykonania. Najbardziej ogólnym obszarem jest cykl życia projektu, dokumentem z nim związanym jest *Plan projektu.* Cykl życia projektu dostarcza

udziałowcom, członkom zespołu widocznych punktów synchronizacji oraz punktów decyzyjnych w całym projekcie.

Proces wytwarzania oprogramowania, zgodnie z OpenUP, podobnie jak RUP, składa się z czterech etapów:

- rozpoczęcia (ang. Inception)
- opracowania (ang. Elaboration)
- budowy (ang. Construction)
- przekazania (ang. Transition).

Etapy kończą się kamieniami milowymi. Każdy z etapów składa się z jednej lub wielu iteracji. Na podstawie dokumentu *Plan projektu* tworzony jest dokument *Plan iteracji.* Artefakt ten związany jest z obszarem OpenUP, jakim jest cyklu życia iteracji. Obszar cykl życia iteracji zapewnia zbiór praktyk opisujących sposób przeprowadzenia iteracji ukierunkowanej na zespół, dostarczającej przyrostowe wartości udziałowcom w przewidywalny sposób. Iteracja wymusza na zespole skoncentrowanie swoich prac na dostarczeniu, co kilka tygodni w pełni przetestowanej, kolejnej części produktu. Przebieg iteracji zgodny z OpenUP przedstawiony jest na rysunku 4.



Rys. 4. Iteracja wg OpenUP

Postęp iteracji widoczny jest przez zakończenie kolejnych mikroprzyrostów (ang. micro-increments). Na początku iteracji organizowane jest spotkanie dotyczące planu iteracji. Spotkanie takie trwa do kilku godzin. Inicjalizacja iteracji trwa od jednego do dwóch dni. Czas ten poświęcony jest na uszczegółowienie planu iteracji, dogłębne zrozumienie zależności logicznych wytwarzanych elementów oraz ich wpływu na architekturę. Większość czasu iteracji poświęcona jest na wykonywanie mikroprzyrostów, które dostarczają przetestowany kod oraz zatwierdzone artefakty. Dla uzyskania większej dyscypliny, pod koniec każdego tygodnia należy zapewnić stabilny kod. Umożliwia to wczesne wykrycie i rozwiązanie problemów. Ostatni tydzień lub kilka ostatnich dni iteracji zazwyczaj poświęconych jest w dużej mierze na poprawę wcześniej zidentyfikowanych błędów. Celem jest dostarczenie, na koniec iteracji, wysokiej jakości produktu zawierającego wymaganą funkcjonalność. Iteracje kończą się oceną produktu, który został podczas niej wytworzony. W ocenie biorą udział zainteresowane strony. Iteracje w różnych etapach różnią się od siebie wykonywanymi czynnościami oraz celami. Cele w poszczególnych fazach są podobne do celów RUP.

Metodyka OpenUP jest lekką metodyką iteracyjnego wytwarzania oprogramowania. Przeznaczona jest dla zespołów składających się z 3-6 sześciu osób oraz projektów, które trwają około sześciu miesięcy. Szczegółowe porównanie RUP i OpenUP zawarto w pracy [5].

5. Zwinny RUP

W projektowaniu systemów informatycznych istotne jest element spełniania wymagań postawionych przez udziałowców.

Z praktyki projektowej wynikają niezbicie dwa fakty. Zbyt sztywne trzymanie się harmonogramu i skupienie na tworzeniu szerokiej listy dokumentów może prowadzić do olbrzymich kosztów i nikłych efektów w sensie wytworzonego produktu końcowego, jakim jest funkcjonujący system informatyczny. Z drugiej strony brak jakiejkolwiek dokumentacji i duża skłonność do wprowadzania zmian na życzenie udziałowców prowadzi do nadmiernego rozprzestrzeniania się zakresu systemu, wielokrotnego przepisywania kodu i jego testowania i w wyniku także wysokich kosztów wytworzenia systemu informatycznego.

W związku z tym istotne jest odpowiednie skonfigurowanie procesu projektowania systemu informatycznego. Kluczowe, zdaniem autora, jest łączenie zalet podejścia zdyscyplinowanego z podejściem zwinnym. Podstawowe jest zastosowanie głównych założeń metodyk, takich jak RUP na poziomie wyższym projektu: etapy. Natomiast na niższym poziomie – iteracji, mikroprzyrostu – należy stosować założenia charakteryzujące metodyki zwinne, takie jak SCRUM i OpenUP.

Do zasad RUP, które warto stosować w projektach informatycznych, należą:

- iteracyjne wytwarzanie
- modelowanie wizualne (UML)
- modelowanie architektury (wybrane widoki)
- zarządzanie wymaganiami w sposób ciągły.

Do zalet RUP należą stosowanie podziału projektu na etapy i iteracje. Zastosowanie czterech etapów z jasno określonymi celami

i kryteriami przejścia do kolejnego etapu przyczynia się do wczesnego zdefiniowania zakresu, weryfikacji architektury, rozwiązania ryzyk projektowych, przewidywalnego dostarczania kolejnych przyrostów oraz przygotowanego przekazania systemu użytkownikowi końcowemu.

Elementem, jaki należy zrealizować przy zastosowaniu RUP w projektach, jest jasne zdefiniowanie zakresu czynności i produktów wchodzących w zakres metodyki stosowanej w określonym projekcie. Podstawowe w RUP jest ograniczenie zbioru wytwarzanych dokumentów oraz modeli do tych naprawdę istotnych w określonym projekcie. Zapewni to odciążenie zespołu projektowego od nadmiernego dokumentowania i modelowania oraz pozwoli skupić się na wytwarzaniu oprogramowania.

Przy projektowaniu systemów informatycznych istotne są kluczowe role definiowane w RUP:

- kierownik projektu
- główny analityk
- główny architekt
- projektant bazy danych
- kierownik testów.

Role te koordynują główne prace w zespole projektowym i mają największy wpływ na kreowanie końcowego rozwiązania, jakim jest system informatyczny. Porównując z metodyką SCRUM, można zauważyć podobieństwo ról kierownika projektu i głównego analityka z RUP oraz mistrza młyna i właściciela produktu ze SCRUM. SCRUM czy OpenUP może wnieść do projektu, paradoksalnie, zdyscyplinowane wytwarzanie praktycznie codziennych mikroprzyrostów oprogramowania. Efektem takiej organizacji pracy jest bardziej stabilne oprogramowanie na koniec iteracji. Należy jednak pamiętać, aby nie popaść w zbyt szczegółowe testowanie poszczególnych mikroprzyrostów. Z drugiej strony testowanie oprogramowania tylko przez tworzących je programistów może prowadzić do pozostawienia w kodzie zbyt wielu ukrytych błędów. Wyjściem jest stosowanie testowania jednostkowego na poziomie przyrostu, a testowania wersji włącznie z testami regresyjnymi na poziomie iteracji.

Kluczowa przy stosowaniu podejścia SCRUM jest znajomość jasno zdefiniowanych wymagań dla przebiegu. W roli właściciela produktu ze SCRUM może występować analityk z RUP zajmujący się określonym wymaganiem funkcjonalnym – przypadkiem użycia. Analityk ściśle współpracuje z członkami zespołu w celu doprecyzowania wymagań realizowanych w przebiegu.

Zasada, aby nie przedłużać czasu trwania iteracji jest wspólna, zarówno dla metodyk RUP, jak i SCRUM. Zarówno RUP, jak i SCRUM podkreślają znaczenie oceny iteracji (przebiegu) dla lepszego dalszego planowania projektu.

Drugim podstawowym elementem procesu wytwórczego jest architektura systemu informatycznego. Jest to element szczególnie podkreślany w metodyce RUP. Metodyka RUP w pełnej postaci zakłada modelowanie systemu informatycznego z punktu widzenia modelu architektonicznego „4+1" [7]. Model ten obejmuje następujące widoki architektoniczne systemu informatycznego: przypadków użycia, logiczny, wdrożeniowy, procesów, implementacyjny.

W większości projektów informatycznych istotne jest modelowanie architektury systemu informatycznego z punktu widzenia następujących widoków architektonicznych modelu „4+1": przypadków użycia, logicznego, wdrożeniowego.

Pierwszy z widoków architektonicznych stanowi widok przypadków użycia. W ramach tego widoku wykonuje się model przypadków użycia. Model ten stanowi specyfikację wymagań na budowany system.

W widoku logicznym przedstawiane są struktura oprogramowania oraz sposób jego działania. Budowany jest w nim model projektowy oraz model bazy danych.

Istotny jest widok wdrożeniowy pokazujący system informatyczny na środowisku uruchomieniowym z fizycznymi serwerami oraz środowiskami uruchomieniowymi. Budowany jest w nim model wdrożeniowy.

Wczesne zdefiniowanie i zweryfikowanie architektury systemu informatycznego zapewnia minimalizację ryzyk projektowych oraz minimalizację kosztów projektu. Zgodnie z RUP, pełen zespół projektowy powoływany jest dopiero w fazie budowy, po wcześniejszej weryfikacji architektury przez niewielki zespół doświadczonych specjalistów.

6. Zwinny RUP w praktyce

Przedstawione powyżej podejście zostało wstępnie zweryfikowane w projekcie uczelnianym systemu informatycznego dla kancelarii prawniczej. Następnie zostało zastosowane w projektach informatycznych istotnie różniących się od siebie:

- projekt dla Służby Zdrowia RP „Modelowanie repozytorium i analiza

efektywności informacyjnej wytycznych i ścieżek klinicznych w służbie zdrowia", nr ref.: POIG.01.03.01-00-145/08, Program Operacyjny „Innowacyjna Gospodarka" – zespół 30 osób

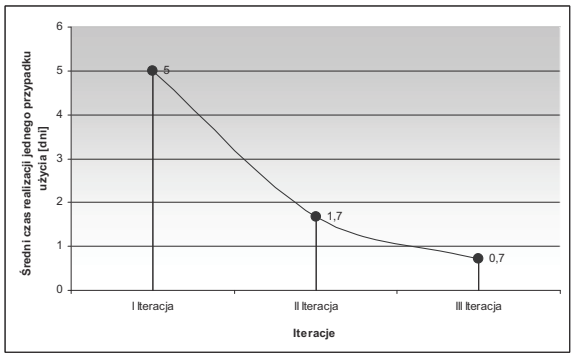
- etap I wdrożenia Zintegrowanego Systemu Wspomagającego Zarządzanie Pojazdami Kolejowymi i Drużynami Trakcyjnymi w PKP CARGO S.A. – zespół 50 osób.

Najistotniejszą cechą wspólną tych projektów informatycznych jest to, że zakończyły się w z góry określonym czasie, dostarczono wymaganą funkcjonalność oraz zrealizowano to w ramach ustalonego wcześniej budżetu.

Dalej przedstawiono analizę wyników budowy fragmentu systemu informatycznego dla kancelarii prawniczej [10]. System ten tworzony był w architekturze Java EE, przy wykorzystaniu dostosowanej konfiguracji RUP. Cały proces składał się z trzech iteracji. W iteracjach zostały uzyskane wartości następujących wskaźników efektywności zespołu projektowego:

- średni czas realizacji przypadku użycia
- liczba zaimplementowanych przypadków użycia w iteracji
- liczba klas wytworzonych w iteracji
- liczba klas ponownie użytych w iteracji
- liczba wszystkich klas użytych w iteracji
- liczba tworzonych metod w iteracji dla realizacji przypadku użycia.

Dzięki wykorzystaniu doświadczenia oraz już istniejących elementów systemu, czas potrzebny na implementacje przypadku użycia w kolejnych iteracjach uległ znacznemu skróceniu. Średni czas realizacji przypadku użycia w kolejnych iteracjach został przedstawiony na rysunku 5.

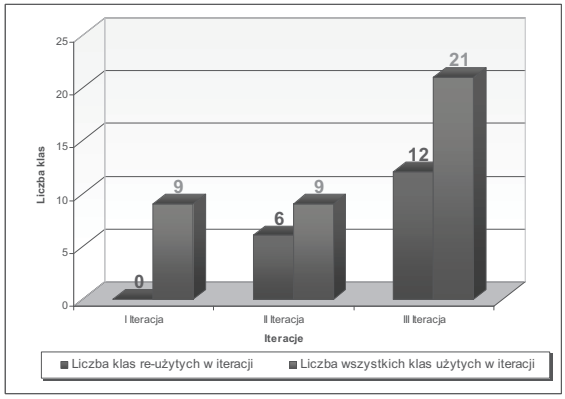


Rys. 5. Wykres średniego czasu realizacji przypadku użycia w iteracji

Każda z iteracji trwała tydzień (pięć dni roboczych). W pierwszej iteracji został zrealizowany jeden przypadek użycia, w drugiej, w tym samym czasie, zaimplementowane zostały trzy przypadki użycia. W trzeciej iteracji liczba wykonanych przypadków użycia wzrosła do siedmiu. Oznacza to, że w tym samym okresie możliwe było dostarczenie większego zakresu funkcjonalności systemu, co przekłada się na wzrost postępu prac i tym samym podniesienie poziomu efektywności zespołu projektowego. Efekt ten wynika:

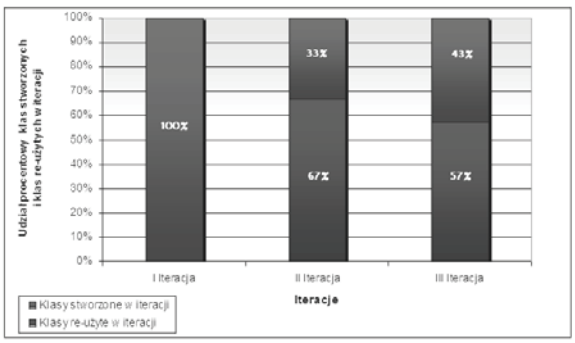
- ze zwiększającego się wskaźnika wtórnego użycia kodu w kolejnych iteracjach
- z nabywania doświadczenia, umiejętności i wiedzy dotyczącej stosowanej technologii
- z wczesnej stabilizacji architektury systemu.

Wraz z postępem prac wzrastała także liczba klas powtórnie użytych w iteracji. W drugiej iteracji liczba ta wyniosła 6, a w trzeciej 12. Rysunek 6 przedstawia wykres liczby klas ponownie użytych oraz wszystkich klas wykorzystanych podczas implementacji w poszczególnych iteracjach.



Rys. 6. Wykres liczby klas ponownie użytych oraz wszystkich klas wykorzystanych w iteracjach

Liczbę klas wytworzonych i powtórnie użytych w iteracji odniesiono także do liczby wszystkich klas użytych w iteracji. Uzyskano procentowy udział obydwu tych liczb w liczbie wszystkich klas użytych w iteracji. Zostało to przedstawione na wykresie na rysunku 7.



Rys. 7. Wykres przedstawiający udział procentowy klas wytworzonych oraz klas ponownie użytych w iteracjach

Ciekawe wyniki uzyskano również, obserwując wskaźnik pokazujący nakład pracy, wyrażony za pomocą liczby tworzonych metod w iteracji dla realizacji przypadku użycia. W pierwszej iteracji należało utworzyć 118 metod, aby zrealizować jeden przypadek użycia. W drugiej − 42 metody, aby zrealizować 3 przypadki użycia, w trzeciej − 71 metod, aby zrealizować 7 przypadków użycia. Liczba tworzonych metod w iteracji dla realizacji przypadku użycia kształtowała się następująco w iteracjach:

- I iteracja − 118 metod
- II iteracja − 14 metod
- III iteracja − 10 metod.

Realizacja przypadku użycia w kolejnych iteracjach odbywała się mniejszym kosztem nakładu prac, m.in. dzięki ponownemu użyciu już istniejącego kodu. W pierwszej iteracji stworzona została podstawowa architektura systemu, co pociąga za sobą potrzebę implementacji klas i ich metod stanowiących rdzeń systemu.

Istotnym aspektem jest fakt, że zastosowanie podejścia iteracyjnego umożliwia wczesne nabywanie doświadczenia i umiejętności, co przekłada się na podniesienie efektywności zespołu projektowego. Ponadto, wcześnie zdefiniowana i zweryfikowana architektura systemu niweluje ryzyko jego technicznej realizacji.

Otrzymane wyniki świadczą jednoznacznie o podnoszeniu w trakcie projektu efektywności zespołu projektowego. Postęp prac zwiększał się w każdej kolejnej iteracji przy jednoczesnym zmniejszaniu się nakładu prac.

7. Podsumowanie

W artykule została zaproponowana metoda iteracyjnej budowy systemów informatycznych łącząca Rational Unified Process i SCRUM. W metodzie tej zaproponowano podejście ukierunkowane na architekturę i łączenie wytwarzania iteracyjnego z zaletami przebiegów SCRUM. Metodyka ta została wykorzystana podczas budowy fragmentu systemu informatycznego wspomagającego pracę kancelarii prawniczej, ale także podczas realizacji ww. projektów komercyjnych. Projekty realizowane zgodnie z przedstawioną metodyką charakteryzowały się następującymi cechami:

- podniesienie przewidywalności projektu
- wczesna stabilizacja architektury systemu informatycznego
- sukcesywne i przewidywalne dostarczanie produktów projektu
- podniesienie ponownego użycia klas w systemie informatycznym
- obniżenie liczby ponownie przepisywanych klas
- skrócony czas realizacji przypadku użycia
- zmniejszenie liczby metod potrzebnych do realizacji przypadku użycia.

Łączenie dwóch elementów: elastycznej organizacji zespołu projektowego oraz ścisłej kontroli architektury systemu informatycznego prowadzi do podnoszenia efektywności zespołu projektowego. Należy podkreślić, że podstawowym widokiem architektonicznym jest widok przypadków użycia. Kluczowe jest utrzymywanie ciągłego porozumienia nad zebranym zestawem wymagań pomiędzy zespołem projektowym a zamawiającym i użytkownikiem końcowym.

8. Bibliografia

[1] J. Barnes, *Implementing the IBM Rational Unified Process and Solutions. A Guide to Improving Your Software Development Capability and Maturity*, IBM Press, 2007.

[2] Dokumentacja OpenUP Version 1.5.0.1.

[3] M. Fowler, *UML Distilled Third Edition*, Addison-Wesley, 2005.

[4] S.H. Kan, *Metryki i modele w inżynierii jakości oprogramowania*, Mikom, 2006.

[5] P. Kroll, B. MacIsaac, *Agility and discipline made easy*, Addison-Wesley, 2006.

[6] P. Kroll, P. Krutchten, *Rational Unified Process od strony praktycznej*, WNT, Warszawa, 2007.

[7] P. Kruchten, *The Rational Unified Process, An Introduction*, Addison-Wesley, USA, 1999.

[8] P. Krutchten, *Rational Unified Process od strony teoretycznej*, WNT, Warszawa, 2007.

[9] *Manifesto for Agile Software Development*, www.agilemanifesto.org.

[10] J. Olszewski, *Metoda iteracyjnej budowy systemów informatycznych w architekturze J2EE*, praca magisterska, kierownik dr inż. Tomasz Górski, WAT, 2009.

[11] *RUP (Rational Unified Process)*, www.javatech.com.pl/rup.html.

[12] Ken Schwaber, *Agile Poject Management with Scrum*, Microsoft Press, Washington, 2004.

[13] *Scrum in five minutes*, www.softhouse.se/Uploades/Scrum_eng_webb.pdf.

[14] Hirotaka Takeuchi, Ikujiro Nonaka, „The New Product Development Game”, *Harvard Business Review*, 01-02.1986.

Agility and discipline in increasing the efficiency of project teams

T. GÓRSKI

The article presents assumptions of the software development methodology which links the features of Rational Unified Process and agile methodologies such as SCRUM and OpenUP. The article presents properties of following methodologies: Rational Unified Process, SCRUM and OpenUP with emphasis on their common features. The article also contains a results analysis of the methodology in a software project in the context of improving the efficiency of the design team working in accordance with this methodology.

Keywords: effectiveness of the software development process, information system architecture